ISSN 0956-9979 MARCH 1994

VIRUS BULLETIN

THE INTERNATIONAL PUBLICATION ON COMPUTER VIRUS PREVENTION, RECOGNITION AND REMOVAL

IN THIS ISSUE:

• **Viruses under Unix.** Why are *Unix* viruses so rare? What is the potential virus threat under *Unix*? Why is the best known virus written in shellscript? Everything you ever wanted to know about computer viruses and their relevance to *Unix* is on page 15.

• **Following the trends.** Along with the Olympic Games comes an Olympic virus, from a new group of Swedish virus authors calling themselves 'Immortal Riot'.

• **Central Point under the microscope.** *CPAV for NetWare* is one of the more popular choices for server-based virus protection, providing a wide range of management features. *VB* takes a look at the package, and finds some good points, some bad.

CONTENTS

EDITORIAL

Zeus with Zits?

One of the benefits of editing a journal is that one gets an almost infinite supply of free books and magazines. This month's batch of light reading contained a copy of the latest *œuvre* from Mark Ludwig, *Computer Viruses, Artificial Life and Evolution.* A review of the book follows next month, but merely from the title, it is clear that Ludwig views viruses as far more than the computer equivalent of a bad attack of acne.

As soon as one looks at what a computer virus does, the similarity to its biological namesake becomes apparent. It replicates, it infects other objects... at first glance, it appears to display many of the characteristics of a primitive life form. Where this comparison is obviously oversimplified, it does raise some questions about the motives of those who trawl the world's Virus Exchange BBSs.

" It would be foolish to suggest that the motivation of all those who collect or distribute viruses is always the same "

It is entirely possible that science can learn from the phenomenon of self-replicating code; viruses may be a legitimate model for some of the processes which we observe in the world around us. The concept of developing viruses which could 'evolve', or simulated competition between computer generated life forms on a computer, is of scientific interest to genuine researchers.

One can see why the possibility (however remote) that viruses might be alive would appeal to a certain mentality. This would mean that each computer programmer has the potential to create life. Is God a spotty teenager - a possibility mysteriously overlooked by contemporary theologians? Will anyone look at a Form-infected diskette in the same light ever again?

For this reason, there is a place for *bona fide* research into computer viruses - not from the point of view of virus prevention, but as an area of study in its own right. Anyone who is interested in the subject is in a dilemma: they cannot study the problem without virus samples. As they are prohibited from obtaining viruses 'above board', they turn to the computer underground in their search for knowledge. Taken at face value, this is a fair stance, but it is completely at odds with the views of the rest of the computer-literate community.

It would be foolish to suggest that the motivation of all those who collect or distribute viruses is the same. Some of these people may be interested in the subject of artificial life. Some might want to understand the subject of computer viruses themselves. Some may simply enjoy being infamous. Whichever way, there will be *some* who have a quasi-legitimate interest in the subject.

Much of the above reads rather like the typical arguments of the virus writers themselves. However, these can be reduced to the following: destruction justified by thirst for knowledge. It should be self-evident to even the most naïve virus writer that anti-virus vendors are forced to update their product for every new virus which is discovered, regardless of whether it is known to be causing a problem 'in the wild'. This adds to development costs and memory overheads.

Even more obvious is that those who release viruses into the wild are either terminally irresponsible or have malicious intent. Either way, such actions effectively nullify any claims to legitimacy.

If those involved in the computer underground have any claims to legitimate research, time for them is running out. Over the next few years, governments around the world will be forced to pass new legislation which will attempt to address the actions of the hacker. These laws may well mark certain subjects 'out of bounds' for many years to come. If any of the virus-writing community truly believe their own arguments about study and research, they should make their views heard quickly - or face even stiffer penalties for their actions.

Knowledge is never intrinsically good or bad; it is the use to which it is put which determines how it should be viewed. However, the place of censorship and virus exchange can only be discussed in an environment where the minority is not imposing its whims on the majority. If the virus writers wish to avoid the restriction of information and the moderating of the *Internet*, they had better take a look at the result of their handiwork: any new restrictive legislation will be their doing.

VIRUS BULLETIN **©1994 Virus Bulletin Ltd, 21 The Quadrant, Abingdon, Oxfordshire, OX14 3YS, England. Tel +44 (0)235 555139. /90/$0.00+2.50**

NEWS

Pathogen Virus Warning

A new virus has recently been discovered 'in the wild' in the UK. The virus is not detected by the latest versions of the scanners from *Frisk Software* (*F-Prot* v2.11), *McAfee Associates* (*SCAN* 9.21 v111), *S&S International* (*Findviru* v6.52) or *Sophos* (*Sweep* v 2.58). It is not known whether any other scanners detect the virus.

The virus is heavily polymorphic, using a highly variable decryption loop. The polymorphic code contains a number of novel features which may make the virus difficult to detect.

On Mondays at 5pm, the virus corrupts random sectors of the hard disk, and displays the following message:

```
Your hard-disk is being corrupted, courtesy of
PATHOGEN!
Programmed in the U.K. (Yes, NOT Bulgaria!)
[C] The Black Baron 1993-4.
Featuring SMEG v0.1:  Simulated Metamorphic
Encryption Generator!
'Smoke me a kipper, I`ll be back for
breakfast.....'
Unfortunately some of your data won`t!!!
```

The virus marks infected files by incrementing the year field by 100. Unfortunately, most standard disk utilities only display the last digits of this field, although it is possible to view this value with a disk editor. This provides an interim detection method until product manufacturers update their scanners. However, due to the obvious trigger message, the long term risk posed by the virus is small.

Internet Break-ins Rise

In the last week in January, *CERT* (the *Computer Emergency Response Team*), based at *Berkeley University* in the United States, observed a dramatic increase in the number of reports of intruders monitoring network traffic. Some systems have been compromised, and those who access remote services via FTP, Telnet, and Rlogin are at risk.

The current attacks take advantage of the promiscuous mode of a specific network interface, /dev/nit, to capture host and user authentication information on newly opened sessions.

As a short term measure, *CERT* has recommended that the /dev/nit feature should be disabled if not used. However, this is not the underlying problem: the fault is not the (in)security of /dev/nit, but the way in which messages are passed around the *Internet* in an unencrypted form, allowing anyone to view the contents of a data packet.

Such loopholes in the security of the *Internet* are a growing problem, as millions of people use the system every day. Attacks of this nature can only be prevented by adopting additional security measures at the packet level.

Virus Prevalence Table - January 1994

Virus	Incidents	(%) Reports
Form	20	46.5%
New Zealand 2	5	11.6%
Spanish Telecom	3	7.0%
Athens	2	4.7%
Parity Boot.A	2	4.7%
3NOP	1	2.3%
4K	1	2.3%
Cascade	1	2.3%
CMOS1	1	2.3%
DIR II	1	2.3%
Lamers Surprise	1	2.3%
Maltese Amoeba	1	2.3%
NoInt	1	2.3%
Parity Boot.B	1	2.3%
Tequila	1	2.3%
Quox	1	2.3%
Total	43	100.0%

Sophos Launches New Product

Oxford-based *Sophos Plc* has launched a long-awaited enhancement to its product range: *InterCheck*, an alternative to the traditional TSR virus scanner.

The new product is aimed at those network users concerned with virus prevention. According to Head of Development, Tim Twaits, *InterCheck* provides true Client-Server virus protection. This is *Sophos'* first foray into the area of TSR virus scanning and checking.

The program automatically maintains a list of authorised executables for each workstation. Whenever a program is run, it is checked against that list. Any attempt to access an unknown or modified program causes the client to request authorisation from the server. Thus, the code which is run on the workstation never needs to be updated, and will not grow as more complex viruses are discovered.

Whether or not the move will pay off depends largely on how it is received in an increasingly complex market. The concept appears to solve some of the problems associated with *virus-specific* TSR virus detection. However, *Virus Bulletin* has yet to test *InterCheck* to determine the performance implications of the product.

Twaits is confident: 'The biggest problem with conventional TSR utilities is their poor performance when checking for highly polymorphic viruses, like MtE or TPE. *InterCheck* avoids this problem.'

VIRUS BULLETIN **©1994 Virus Bulletin Ltd, 21 The Quadrant, Abingdon, Oxfordshire, OX14 3YS, England. Tel +44 (0)235 555139. /90/$0.00+2.50**

IBM PC VIRUSES (UPDATE)

The following is a list of updates and amendments to the *Virus Bulletin Table of Known IBM PC Viruses* as of 17 February 1994. Each entry consists of the virus name, its aliases (if any) and the virus type. This is followed by a short description (if available) and a 24-byte hexadecimal search pattern to detect the presence of the virus with a disk utility or a dedicated scanner which contains a user-updatable pattern library.

Type Codes

- **C** Infects COM files
- **D** Infects DOS Boot Sector (logical sector 0 on disk)
- **E** Infects EXE files
- **L** Link virus
- **M** Infects Master Boot Sector (Track 0, Head 0, Sector 1)
- **N** Not memory-resident
- **P** Companion virus
- **R** Memory-resident after infection

Abbas

CER: This 1320-byte virus is also known as 'Iranian'. Awaiting analysis.

```
Abbas           3D00 4B75 612E 8C1E C501 2E89 16C7 0132 C0E8 B902 2E89 0EC9
```

Abraxas

EN: Two new variants of this overwriting virus, 1171 and 1200 bytes long.

```
Abraxas.1171    CD21 B43C 33C9 BA9E 00CD 21B7 4093 BA00 01B9 9304 CD21 C3B4
Abraxas.1200    CD21 B43C 33C9 BA9E 00CD 21B7 4093 BA00 01B9 B004 CD21 C3B4
```

Anti-MIT.764

CN: This variant does not seem to work properly, and should probably be classified as 'intended'.

```
Anti-MIT.764    BE27 018A 2605 01EB 11AC 32C4 AAE2 FAB4 19CD 218A F0B4 0ECD
```

ARCV

Three new variants have appeared, but it seems they are created by making slight modifications to older variants. ARCV.Ice-9.642 (CN) is detected with the Ice-9 pattern, ARCV.Jo.912 (CR) is closely related to the the Jo.911 variant, and ARCV.Anna.745 (CN) is derived from a 742-byte variant.

```
ARCV.Jo.912     BE?? ??B9 BD01 2E81 ???? ??83 C602 4975 F5
ARCV.Anna.745   8DBC 1D01 B9A8 0280 35?? 47E2 FAC3 FE84 CE03 E8D6 FFE8 E8FF
```

Armagedon.1079.E

CR: A minor variant, detected with the Armagedon pattern.

Ash

CN: Four new variants of the Ash virus have appeared recently. One (Ash.1604) is detected with the pattern published for Ash.1602, but the other three are new. The 441- and 451-byte variants seem to be created from the same source, but assembled using different assemblers. According to the source code the author's name for the original Ash virus was 'Born on the 4th of July'.

```
Ash.737         E800 005D 81ED 0B01 8D9E 2B01 533E 8A86 2301 B9BA 0230 0743
Ash.441         8DB6 0501 BF00 01B9 0400 FCF3 A4B4 1A8D 96BE 02CD 21B4 4E8D
Ash.451         8DB6 0501 BF00 01B9 0400 FCF3 A4B4 1A8D 96C8 02CD 21B4 4E8D
```

AT.140.B

CR: New variant, not significantly different from the 'A' variant.

```
AT.140.B        C9B8 0042 CDF7 B440 8D54 FF89 2CB1 03CD F7B4 3ECD F71F 61EA
```

BA

CN: A simple, 181-byte virus with no payload. It uses the letters 'BA' - possibly the author's initials - to mark files as infected.

```
BA              817E 0342 4174 41BB 8000 8B57 1A81 C2B5 0081 C200 0189 1606
```

BadSectors

CER: This 3428-byte variant contains the text 'Badsectors 1.2', which might indicate that versions 1.0 and 1.1 exist as well. Awaiting analysis, but reported to be 'in the wild' in Israel.

```
BadSectors      3D00 4B75 03E9 6109 3D00 3D75 03E9 5909 80FC 4E75 03E9 AF09
```

Beer.2984

CER: Yet another Russian Beer variant.

```
Beer.2984       FA90 80FC 3B75 03E9 72FF 3D00 3D74 0F3D 023D 740A 80FC 5674
```

Black_Jec

CN: Three new variants have been found: 230, 246 and Sad.300. They are all detected with the Black_Jec (Bljec) pattern.

Burger

CN: Eight variants of this overwriting virus have not been mentioned before, but are all detected with the 'Burger' pattern. They are 405.D, 405.E, 441, 512, 560.AJ, 560.AK, 560.M, 560.X.

Burma

CEN: The virus that was originally reported as 'Burma' has now been renamed Burma.442.A, and two new variants have been found, 442 and 563 bytes long. Both overwrite files they infect and are therefore extremely unlikely to spread.

```
Burma.442.B     2E01 E8F9 00E8 CE00 E8D3 00E8 F000 E814 01E8 CA00 E819 01E8
Burma.563       3101 E869 01E8 FF00 E863 01E8 F900 E869 01E8 1F01 E8E0 00E8
```

Butterfly.FJM

CN: A new variant with the text string changed. It is 302 bytes long like the original, and detected with the Butterfly pattern [*for a full analysis of the Butterfly virus, see p.8. Ed.*].

Cascade

CR: The new Cascade variants this month are 1699, 1701.N (detected with the Cascade(1) pattern), 1701.Jojo.F (detected with the Jojo pattern) and 1702.

```
Cascade.1699    012E F687 2A01 0174 0F8D B74D 01BC 8006 3134 3124 464C 75F8
Cascade.1702    012E F687 2A01 0174 0F8D B74D 01BC 8306 3134 3124 464C 75F8
```

Clonewar

P: Three new viruses belonging to this family of small companion viruses.

```
Clonewar.228    93B9 E400 BA00 01B4 40CD 21B4 3ECD 21BA 1901 B903 00B8 0143
Clonewar.246    8BD8 B9F6 00BA 0001 B440 CD21 B43E CD21 BA1A 01B9 0300 B801
Clonewar.261    8BD8 B905 01BA 0001 B440 CD21 B43E CD21 BA28 01B9 0300 B801
```

Danish_Tiny

CN: Three new variants have been found recently - a 308-byte variant which does not work properly, and two partially encrypted variants, 311 and 476 bytes long. The 476-byte variant was discovered 'in the wild' in Estonia.

```
Danish_Tiny.308 8BD7 B902 00B4 3FCD 2181 3D07 0874 D6B4 2CCD 210B D274 F889
Danish_Tiny.311 AD33 C3AB E2FA 5E59 5B58 C3E8 DBFF 8984 3F02 B440 8D94 0501
Danish_Tiny.476 AD33 C3AB E2FA 5B59 585E C3E8 DCFF 8984 E402 8D94 0501 B440
```

Dark_Avenger

CER: Two new Dark_Avenger variants are now known, 1800.K and Major (1832 bytes). They are both detected with the Dark_Avenger search string.

Deicide II

CN: Four new variants, all of which have been created by making slight modifications to older variants.

```
Deicide_II.359  B440 BA00 01B9 4C01 CD21 B457 B001 5A59 CD21 B43E CD21 8B1E
Deicide_II.622  B440 BA00 01B9 5202 CD21 B457 B001 5A59 CD21 B43E CD21 8B1E
Deicide_II.623  B440 BA00 01B9 5302 CD21 B457 B001 5A59 CD21 B43E CD21 8B1E
Deicide_II.240  B440 BA00 01B9 4909 CD21 B457 B001 5A59 CD21 B43E CD21 8B1E
```

Dutch_Tiny.111

CN: An unremarkable, 111-byte variant.

```
Dutch_Tiny.111  93B4 3FCD 2180 3C4D 741C B002 E826 0097 B16F B440 CD21 B000
```

Flash.688.C

CER: A minor variant, possibly changed to avoid detection using the *Virus Bulletin* search string.

```
Flash.688.C     005E 8BDE 81C3 0F00 B000 D50A FA88 07EB 05EA ???? ???? FBC6
```

Hates.212

CN: Very similar to the original Hates virus, and detected with the same pattern. This virus is 212 bytes long, one byte shorter than the original.

HH&H.4087

CR: Similar to other known variants, but of a different size.

```
HH&H.4087       50B9 F70F 8B1E 0101 81C3 1501 8037 ??43 E2FA
```

Infector

CN: Several new variants of this virus, which is almost certainly of Russian origin.

```
Infector.608    A200 01A0 EE02 2EA2 0101 A0EF 022E A202 01B9 7F00 BB81 002E
Infector.692    A200 01A0 2F03 2EA2 0101 A030 032E A202 01B9 9000 BB00 002E
Infector.695    A200 01A0 4503 2EA2 0101 A046 032E A202 01B9 7F00 BB81 002E
Infector.752    A200 01A0 FC03 2EA2 0101 A0FD 032E A202 011E 0633 C08B F08E
```

Keypress.1232.L

CER: Yet another minor variant, detected with the Keypress search string.

Peter

CR: Two new variants are known, B and C. They are 529 bytes like the original, and detected with the same pattern.

PS-MPC

Not surprisingly, there are several new PS-MPC-generated viruses this month, including: 352 (CN), 432 (CN), 574.B (CEN), 603 (CEN), 605 (CEN), 607 (CEN), 611.A (CEN), 611.B (CEN), Seven_Percent (672 bytes CER), 783 (CN) and Swansong.1521. Anti-virus programs capable of detecting PS-MPC viruses should be able to detect all of these variants.

Tiny_Family.Fred

CER: Unlike most of the other members of this family, this variant did not originate in Bulgaria, but rather in Italy. It is 255 bytes long, contains the word 'Fred', and seems to do nothing but replicate.

```
Tiny.Fred       268F 851C 01AB 8CC8 E2F3 2EFF 361D 062E FF36 1F06 BF00 018F
```

VCL

This month brings the following variants: Divide.554 (overwriting), Diarrhea.931 (CN), Eddie (1019 bytes, CN) and Olympic (1440, CN) The Olympic variant seems to have been changed somewhat, probably to avoid some unknown scanner [*for a full analysis of the Olympic virus, see p.9. Ed.*]. The other variants are regular VCL-generated viruses, and should be detectable by any program which is capable of generic detection of VCL variants.

Wilbur.D

CN: Detected with the Wilbur.B pattern.

Wordswap.1085.B

CER: Detected with the Wordswap (Words) pattern.

INSIGHT

Viruses the Whitbread Way

The UK company *Whitbread Plc* is probably best known for its beer, food and leisure concepts. Ian Carman has been with *Whitbread* for almost sixteen years, during which time the company's IT requirements have changed beyond recognition. As an IT Manager who has survived the comings and goings of many different fads and phases during the industry's adolescence, he has seen computer viruses grow from a curiosity into a real business issue.

Preceding the Problem

When Carman first saw what computer viruses might be capable of, he realised that it would be the habits of the user which would be critical; if those could be controlled, the problem would be minimised. Has this approach worked? 'On the whole, our record here has been pretty good - last year, we brought the number of incidents down by about 70%, and I am convinced that it is a result of good practices: the easy things. People might get bored with me going on about viruses left, right and centre. They can recite the virus guidelines word for word, but it works,' says Carman.

One of Carman's many different responsibilities is drafting the company's IT security policy, which he finds himself simplifying year by year. 'The security policy gets thinner annually, every time I review it. I am going through the exercise at the moment. It starts off with a full policy statement which says that we will protect our information assets to maximise their availability, integrity, and confidentiality, commensurate to their value to the business.'

'This is backed up by a statement saying that we will stand by any appropriate legislation, with explicit note made of the *Data Protection Act*, *Copyright Designs and Patents Act* and the *Computer Misuse Act*. There is a little bit about PC viruses and how to avoid them by using only approved software. The stable of products used is getting smaller, and there is less and less need for people to get out and do their own thing. That's really all that the policy contains.' Carman's strategy is simple, but effective: 'The more concise you make the policy, the more likely it is that people will read it and remember it.'

The Carrot and the Stick

The simplification of the IT policy has made life easier for users, and has vastly reduced the number of virus outbreaks within *Whitbread*. '1991 and 1992 were the worst years - we had around a dozen virus outbreaks. That was partly because we were beginning to scan disks as a matter of course - no longer just responding to users who had a problem. Therefore the incident rate went up, stayed level for those two years, and then came down quite dramatically in 1993. I think that is because we have spent quite a substantial amount of time promoting good practices, often with one-liners: "don't use unapproved software", "scan all incoming disks"... It is the simple things which I believe have kept us relatively virus-free. We were down to only three or four incidents last year.'

Whitbread's approach of simplifying the policy as much as possible is reinforced by the threat of disciplinary action. 'There is a statement within our company policy that the IT policy is mandatory, and failure to comply will render you liable to disciplinary action. Having said that, I think that each incident is treated on its merits. If somebody came to me and said that they had a problem with their machine, and that problem turned out to be virus-related, then the division concerned would have to decide what to do next. In some cases further action would be appropriate, and in some it would certainly not be. It is a tricky one - the policy allows for disciplinary action to be taken; from there on, it really is a matter of discretion. I think this is a good thing - one can have too much management from the centre.'

Backup and Recovery

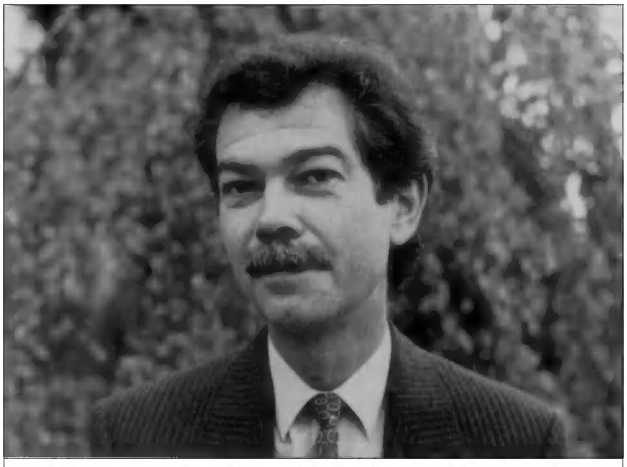
One of the most unforgettable incidents underlined a vital lesson to all in *Whitbread's* IT community. 'The worst incident we ever had was when an engineer brought in a scanner, loaded it up, and we sat and watched it delete everything - we then discovered that it was infected with Dark Avenger. He was terribly contrite, as well he might have been. Basically, he had forgotten to keep his disk write-protected, had been elsewhere, and then brought it here,' said Carman.

'That situation was actually easier to deal with than the sort of niggling viruses which we have had, like Form or New Zealand II. We had carried out a full system backup the previous night, and so we simply decided to wipe the lot and start again - we had the system up and running again within a few hours.'

'The Dark Avenger incident taught us a number of valuable lessons. Most importantly, it showed us the value of backup and recovery techniques. However, more than this, it underlined the value of *frequent* backups - because we had backed up the file server the previous evening, we could simply wipe the disk and start again. Whether those lessons are still as clear as they could be in the users' minds, I somehow doubt.'

Legal Issues

Carman believes that viruses are more than just another hurdle for beleaguered IT Managers: 'It is a business problem, but also a legal one, if only because nobody is quite



Carman: 'last year, we brought the number of incidents down by about 70%, and I am convinced that it is a result of good practices: the easy things.'

sure where virus writers stand in law, and what their rights are.' This area is likely to become more of an issue as time progresses, and needs to be dealt with soon.

However, the only way forwards is through the law: 'If you are going to do something about the virus authors, about transmitting viruses from one company to another, then I think it has to come within the realms of the law.'

Unfortunately, life is, as always, more complicated. 'I'm not sure that introducing legislation, criminal or otherwise, would actually help reduce the number of viruses which appear. How many virus authors are you actually going to get hold of ? How many viruses are you going to prevent? The things are there; some of them have been around for a good many years. They're not going to go away, and legislation won't change that,' said Carman.

Does he think that censorship of virus code would be of use? 'I am fairly passionately opposed to Draconian censorship on the grounds that it might actually cause someone a problem. I think the legal and moral issues are rather different, but at some point someone is going to have to try and address it.'

Connecting into the 90s

Carman believes that new issues within the industry will arise, due to the growing interconnectivity between and within companies. As more users work on home PCs, and as inter-platform connections grow, the job will be one of maintaining the integrity of the data on a number of different platforms. He hopes that the problem can be tackled by an extension of his current policies, building on the already high level of user awareness.

'The number of users who are now starting to buy home PCs is rising - there is a disk traffic which we are just beginning to experience,' explained Carman. This clearly increases the opportunity for a virus to be introduced into the system. However, he is not too concerned: 'We've overcome a lot of hurdles over the years,' he said, 'and we'll overcome any problems the future may bring.'

VIRUS ANALYSIS 1

Goddamn Butterflies!

Jim Bates

I still receive code samples via the twin scourge of Virus Exchange BBSs and irresponsible individuals who write so-called 'research' viruses. The specimen this month is a case in point: despite the fact that the code is small and extremely primitive, it represents another threat with which anti-virus programs must contend.

The halcyon days of virus-free computing are long gone, but the problem could be contained if the pool of existing viruses were not constantly being replenished. Whether the 'Butterfly' virus was written for research, or is just another example of malevolence or irresponsibility, is beside the point. The plain fact is that it - and others like it which continue to pour in to investigators - simply increases the existing problems.

This virus is extremely simple; a one-shot infector of COM files (i.e. non-resident) which appends its 302-byte code to suitable targets, modifying the initial program to ensure that the virus code gets executed first. There is neither a destructive trigger routine, nor are there any overt messages, although the text 'Goddamn Butterflies' is visible as plain text within the code. There are some signs that additional routines were considered or discarded, but in general the code is quite straightforward, and unlikely to cause problems for most anti-virus software. The virus' only attempt at concealment is preserving the date and time of infected files.

Operation and Infection

One of the problems facing virus writers is to make the code determine its own location in memory so that internal data areas can be accessed. This arises because appending parasitic virus code is invariably attached to files of different sizes, and can therefore be loaded anywhere within a specific segment of memory.

Self-location is usually done by stack manipulation, followed by creation of a reference point. This is usually held in a register, but is sometimes stored as data. Butterfly is no different in this respect, and its reference location is maintained in a single register at all times.

Once the reference point has been established, the original four bytes of the host program are copied back to the beginning of the file memory image. When this operation has been completed, the virus begins its search for other files to infect. First, a counter is set to zero. Then, a search request is issued to DOS for files in the current directory which have a COM extension. This initial search includes files with attributes of Read Only, System and Hidden. If no matching files are found, processing will return immediately to the host program.

When a matching file is found, it is opened, and the first four bytes are read into the virus data area. The targeted file is then checked to see whether the letters 'ND' are the sixth and seventh letters of the file name. If so, the file is closed, and processing moves back to look for another matching file. This prevents the virus from infecting the usual DOS command interpreter (COMMAND.COM), which would cause a fairly swift system malfunction.

The next check consists of examining the fourth byte of the file (read into memory after the initial search). If the value 01h is found in this position, the file is considered already infected; it is closed, and processing jumps back to continue the search. The next test checks the size of the target file. If it is not between 121 and 64768 bytes long, it is deemed unsuitable, and is rejected.

Once a file has been found suitable for infection, the original four bytes from the beginning of the file are appended to its end, along with 298 bytes of virus code. Finally, the virus calculates the necessary jump distance and writes the three-byte jump instruction, together with the single infection indicator byte (of value 01h) to the beginning of the file, overwriting the original four bytes. The search routines are extremely primitive and fetch only those files which are in the current directory.

Once the infection process is complete, the counter (noted above) is incremented and checked to see whether it has reached a value of four. If not, a further search is made; otherwise, processing returns to the host program.

Comments

This sample program has been downloaded from a Bulletin Board in Italy (due to the introduction of a new law in that country, virus exchange there is now illegal). When executed, the host program produces this text on screen:

```
                ATTENZIONE!!!
 Questo File è infettato dal virus Butterflies
                ^^^^^^^^^^^
Il file è stato prelevato dall'area Virus di
Euforia BBs, a cuipossono accedere solo utenti
abilitati, secondo quanto descritto nel
Regolamento dell'area stessa.

Questo virus è reso disponibile UNICAMENTE a
scopo di studio. Per eventuali consigli o
delucidazioni rivolgersi a:
XXXXXXX BBS 24h/24 Tel XXXXXXXXXXX (AreaVirus)
```

[*The name and telephone number of the BBS have not been reproduced above. Ed.*]

The text is stored in encrypted form and is decrypted by a simple algorithm when the program is executed. Quite why this is so, and what the text means, is a mystery to me, but it

is plainly an indication that the file came from a Virus Exchange BBS. It is possible that the text is a warning: in that case, it comes too late, since it would not be displayed until *after* the included virus code had been executed (infecting up to three files in the process).

Conclusions

The proliferation of Virus Exchange Bulletin Boards, like virus writing itself, is a small part of a much larger problem. The global interconnection of personal computers has enabled information of all kinds to be made available to anyone with a small amount of time and money to spare. The information alone has no intrinsic value; it only gains value from usage, and that usage can be for good or evil.

The virus writer or exchange BBS operator will happily propagate malicious virus code (under whatever guise) and then complain when others attempt to protect themselves against these actions. Information in the wrong hands can be deadly, causing anything from minor commercial loss to serious damage and death. There is no easy answer, other than educating people to adopt a balanced point of view, accepting that with each right comes an equal obligation.

If an individual wants to exercise his right to become involved in anti-virus research, then he must accept an obligation to prevent the proliferation of virus information. Such obligations and responsibilities should form an integral part of all computing activities under the general title of Computer Ethics. Anyone not subscribing to this view should be excluded from interacting with the exciting world which is computing.

Butterfly

Aliases:	None known.
Type:	Non-resident, parasitic, appending.
Infection:	All COM files between 121 and 64768 bytes long in the current directory (excluding COMMAND.COM).
Self-recognition in Memory:	None necessary.
Self-recognition in Files:	Fourth byte has a value of 1.
Hex Pattern:	`E800 005D 81ED 0B01 BF00 018D` `B604 01B9 0400 FCF3 A4B4 1A8D`
Intercepts:	None.
Trigger:	None.
Removal:	Replace infected files with clean originals. Specific disinfection possible under controlled conditions.

VIRUS ANALYSIS 2

Olympic Games

Mikko Hypponen
Data Fellows, Finland

A new virus, known as Olympic (aka Olympic Aids), has featured prominently on the television, on the radio, and in the newspapers of Northern Europe since the beginning of February. Its newsworthy factors are its Olympic-theme activation routine, and suspicions that it had infected the computer systems of the Lillehammer 1994 Winter Olympics. Fortunately this was not the case.

Despite being reported in the wild in Norway, Olympic is not of Norwegian origin: it is made in Sweden by a new virus group which calls itself 'Immortal Riot'.

Into the Underground

Swedish soil seems to provide particularly fertile ground for raising virus groups: clans like Beta Boys, Demoralized Youth, and the Funky Pack of Cyber Punks have been active in Sweden in the past. The latest group of virus writers, Immortal Riot, seems to consist of four members, known only by their aliases, or 'handles'. So far, the group has published and distributed about thirty viruses, most of which are new variants of existing strains. The viruses thus far seen are not examples of technical brilliance; quite the opposite. Most simply crash the computer, or manifest their presence in some other obvious way.

Immortal Riot also publishes an electronic magazine, Insane Reality, containing articles by the group members and their associates, source codes of viruses, and back-patting and back-stabbing of other members of the virus community. The group seems to be little more than an ego trip for this gang of teenagers - it seems to be 'cool' to be a virus writer.

The computer underground gets steadily more organised - Immortal Riot even publishes its own electronic magazine.

Virus Operation

Olympic is a fairly typical COM file infector, which does not remain in memory, and spreads only when an infected file is executed. Its method of searching for files for infection is not very efficient. Once a number of files on the hard disk have been infected, it may take half a minute to find a new victim: such a slowdown is likely to make the virus easier to spot.

When it finds a suitable candidate for infection, the virus first checks the size of that file to ensure that the infected code will be greater than 64 Kbytes, the largest permissible size for a COM file. The first bytes of the file are checked for a jump construct which the virus is about to insert. If found, the virus considers the file already infected and starts to search for another victim. This process is repeated until five files are infected.

The virus does not check the internal structure of the host file when it infects. Thus, EXE files with a COM extension will be infected by the virus. When such a corrupted file is executed, the virus will infects other files on the machine, but is unable to return control to the original program. In most cases, the machine will crash.

The infection process consists of storing the original first three bytes of the file at the file end, replacing them with a jump to a setup routine, which the virus adds to the end of the file. An encrypted version of the virus code is appended to the end of the file, and, finally, the virus adds a short plain-text note and the decryption routine.

Olympic uses a single pseudo-random variable key based on infection time to encrypt its code. The routine uses either the SI or DI register as work-registers in the decryption loop, alternating between infections. Thus, there are only 25 constant bytes between different virus generations. These are located in two different parts of the virus. The encryption method is not truly polymorphic, and is unlikely to cause problems for anti-virus vendors.

Olympic can infect files which have the DOS Read-Only attribute turned on, and will also restore the date and time stamps of infected files. However, files grow in size by 1440 bytes, which is visible in the directory listing. The virus has no directory-stealth routines, as it does not stay resident.

Olmypian Trigger

The virus was programmed to trigger on the day after the start of the 1994 Winter Olympics (12 February), and has a one-in-ten chance of activating after this date. 'Dice-throwing' is done by checking whether the system timer's hundredth-of-seconds field is below 10. The virus does not check the current year. If the trigger conditions are not met, the virus returns control to the host file.

On activation, the virus draws the Olympic circles on the screen, displaying comments on the Games and its mascots, Haakon and Kristin. Next, it overwrites the first 256 sectors of the first hard disk in the system. To ensure destruction, the virus disables Ctrl-C and Ctrl-Break checking during the destruction routine. Finally, the machine hangs.

While the virus overwrites the fixed disk, it thoughtfully displays the Olympic rings - a symbol of cooperation and unity.

Much of Olympic's code resembles that of viruses generated with VCL, up to the point of the standard VCL-like note; a short message in the end of the virus, which is not displayed at all. The virus' note text reads: 'Olympic Aid(s) `94 (c) The Penetrator'. This virus is probably based on VCL-created code, modified to avoid detection by some scanners. As the virus displays a picture before starting to overwrite the disk, aware computer users might be able to switch the machine off before the virus has a chance to overwrite data areas, making recovery much easier.

VCL.Olympic

Aliases:	Olympic Aids.
Type:	Non-resident, parasitic.
Infection:	Files with 'COM' extension.
Self-recognition in Files:	File starts with a JMP to an offset 1443h from the file end.
Hex Pattern:	Due to the short length and large amount of wildcards, this searchstring should be used with care.

```
8D?? 1301 B9AC 0281 ???? ????
??E2 F8C3
```

Intercepts:	None.
Trigger:	One in ten chance of overwriting the contents of the fixed disk, on or after 12 February, any year.
Removal:	Specific and generic removal possible under clean system conditions. Recovery of machines affected by trigger routine might be possible with specialist data recovery equipment.

VIRUS ANALYSIS 3

Shifting Objectives

Eugene Kaspersky

One of the constant features of all viruses to date is that they must infect executable code. However, a new virus sent to me in January breaks this rule: Shifting Objectives does not infect executable files, but it is perfectly capable of spreading. The virus' infection target is Object modules (OBJ files). These files cannot be executed *per se*, but the virus can spread once such a file has been linked to form an executable file. In order to understand how the virus accomplishes this feat, we must first examine OBJ files in more detail, and gain an understanding of how programs are written and linked.

Inside the Object File

Shifting Objectives spreads by taking advantage of the internal structure of Object files. Most ordinary end-users would probably not know what an Object module is, because DOS programs are usually distributed in their executable form: this usually means COM, EXE or SYS files (often combined with a simple batch file to make the user's life more simple). A brief guide to Object files and their function now follows, as without it, it is impossible to explain how the virus works.

Object files are an intermediate point between source code (the computer language in which the programmer works) and executable code (the binary instructions executed by the computer). Most computer programmers write their code in high level language (eg. C, C++, Pascal), or assembler. This source code is then compiled into one or more Object files, which are subsequently linked to form an executable file. If the program needs to use library functions, these are linked in at the Object file stage.

Any object module is a sequence of variable length object records. This is shown below:

1st Record	2nd Record		...		Last Record

Each record begins with a one-byte field, which specifies the type of record. This byte is followed by a two-byte field containing the length of the remainder of the record (in bytes). Next comes the variable length information field, which contains the binary image of code and data, external references, external and internal names, address references, debugging and miscellaneous information. The very last byte of the object record is a checksum of itself. This means that each record contains the following information:

Type	Length	Information Field	Sum

The main types of object record are:

- 80h - Translator Header Record
- 8Ah - Module End Record
- 8Ch - External Names Definition Record
- A0h - Logical Enumerated Data Record
- A2h - Logical Iterated Data Record

A typical Object module begins with a Translator Header Record (type 80h), and ends with a Module End Record (type 8Ah). These records contain the name of the Object module, the addresses of the main routine and its entrance point (if required).

External Names Definition Record (type 8Ch) contains a list of external names and name types of code and data defined in other Object modules.

Logical Data Records (type A0h and A2h) contain contiguous binary data, or iterated patterns (caused by the presence of a DUP instruction in the source code). That data is either executable code or program data. The object record also contains two other fields: the segment index and data offset. The segment index is the number of segments of the file into which the data will be placed. The data offset indicates the location in segment where data is to be placed.

How it Replicates

Now that we have a clear idea of the information contained within an Object file, it is easy to see how it could become a target for virus infection. Once an Object file is infected, the virus cannot spread until the moment it is linked to form an executable file.

Shifting Objectives tries to infect Object files which are to be used to form a COM file. The virus inserts itself into the Object file so that after linking, the COM file starts with the virus code. Therefore, when this file is executed, control is passed to the virus, which becomes memory-resident.

When an infected file is run, the virus first checks whether it is already installed in memory, by using an 'Are you there?' call. This consists of calling Int 21h with the value FEADh loaded into the AX register.

If this call returns with the value D00Dh in AX, the virus assumes that it is already active in memory, and another call is made to the memory-resident part of the virus code. This consists of another Int 21h call, with the value DEADh in the AX register. This call causes the TSR copy of the virus to disinfect the memory image of the COM file: the virus shifts the host code back to its original location in memory, and returns control to it.

If the 'Are you there?' call goes unanswered, the virus decreases the RAM size of the computer, corrects the last MCB (Memory Control Block), installs itself to the top of memory and hooks Int 21h. The virus then restores the memory image of the host program and passes control to it.

Infection of Object Files

Once memory-resident, the virus intercepts three different subfunctions of Int 21h: FEh, used for an 'Are you there?' call, DEh, used for host program reconstruction, and 3Eh (Close_File), used for file infection. Whenever a file is closed, Shifting Objectives checks the file's extension, using undocumented system file tables. If the extension is OBJ, the virus attempts to infect the file.

The virus reads the first three bytes of each object record within the file. These three bytes contain the record type and length. The virus checks the record type, and if a module is either the Module End Record (type 8Ah), External Names Definition Record (type 8Ch), Logical Data Record (type A0h or A2h), the virus infection procedure calls the corresponding infection routine. In any other case, the virus seeks to the next record.

"Shifting Objectives does not infect executable files, but is perfectly capable of spreading"

In the case of a Data Record (type A0h or A2h), the virus alters its data offset, adding its length in COM files (983h bytes) to that offset. The virus then calculates the new checksum of the record and alters the checksum field, as well as the data offset field of the object record. As a result, all data records will have new data offsets after infection, and all binary data will be offset by 983h bytes on linking.

Shifting Objectives pays particular attention to the first Data Record of the OBJ file. If its data offset is equal to 0100h (this would be the case for an Object file destined to become a COM file), the file is deemed suitable for infection. If the offset is not 0100h (that is, the Object file is not intended to be used to form a host in the COM file format), the virus aborts its infection routine and returns control to the original DOS Int 21h handler.

When the virus has completed this process, the data offset of first data record of infected OBJ files is 04D7h (that is, 0100h+983h, offset 0100h plus the length of the virus code). As a result, OBJ files are not infected twice by the virus: it requires that infected files have an offset of 0100h.

If the next record is a Module End Record (type 8Ah), the virus reads this record into its internal buffer and writes a new Data Record rather than the original Module End Record. This new Data Record contains the virus code with data offset 0100h, so on linking, that record will be placed at the file beginning. The virus then writes the original Module End Record at the end of the Object file.

Trigger Routine

If the virus finds an External Names Definition Record (type 8Ch), it checks the system timer (the word at the address 0000:046C). If the two lowest bits are zero, the virus calls its trigger routine. That routine shifts the screen and displays the message:

```
Shifting Objective .OBJ Virus (c) 1993 by
Stormbringer Kudos for The Nightmare for his
ideas and coolness.
Greets go out to Phalcon/Skism, Urnst Kouch,
Mark Ludwig, NuKE, and everyone else in the
community.
```

The virus then waits for a keystroke, on which it returns control to infection routine.

Problems and Possibilities

The virus has a bug which can cause irreparable damage to some files linked using infected Object files. Shifting Objectives is capable of functioning correctly only from a host which is in the COM file format. However, there is nothing in an Object file which proves that it is intended for use as a COM file. The virus decides whether to infect a particular file by examining the data offset field of the first Data Record. There is no reason why this cannot be 0100h in the case of an OBJ file intended to create an EXE file. This is a fundamental flaw in the virus.

Even with this in mind, Shifting Objectives does pose rather a problem for those marketing generic virus detection packages. Traditional generic techniques rely on the virus carrying out an operation which reveals its presence, such as opening an executable file with read/write access. This virus will circumvent many of these checks, potentially making it difficult to prevent.

Shifting Objectives

Aliases:	SOBJ.
Type:	Memory-resident parasitic file infector.
Infection:	OBJ files only.
Self-recognition in Files:	See text.
Self-recognition in Memory:	'Are you here?' call consists of Int 21h with AX=FEADh. Checks for a return value of D00Dh in AX.
Hex Pattern:	B8AD FECD 213D 0D0D 7503 EB54 908C D848 8ED8 812E 0300 8000
Intercepts:	Int 21h for infection and trigger routine.
Trigger:	Shifts the screen and displays message.
Removal:	Under clean system conditions identify and replace infected files or recompile the sources.

TECHNICAL NOTES

Virus Testing and Identification

Fridrik Skulason

What is a Virus Test Centre (VTC)? The name is somewhat misleading, as the main function of a VTC is probably not to test viruses - any testing probably involves testing of anti-virus programs, in particular: scanners.

Producers and users of anti-virus products generally agree on the need for an independent test centre, but unfortunately no organisation exists today which is able to do this job properly, although some are better than others.

Running a VTC is more complicated than merely collecting a few thousand viruses, running several scanners on that collection, and reporting the results. Many current (and past) VTCs have not been able to fulfil their original intention, but the reasons for their failure may differ. There are myriad grounds why a VTC might fail: some of the more common problems facing the would-be reviewer are listed below.

Insufficient Resources: Doing serious full-scale testing of anti-virus products, as well as maintaining a virus collection, is not something which one person can do as a part-time job: a dedicated full-time researcher with a couple of part-time assistants is required. The basic cost of running even a modest test centre is around £50,000 per year: where will that money originate?

It might be possible to obtain grants from the government or from a computer industry organization, but frequently the VTC either charges the manufacturer for testing, or charges possible buyers for the test results. This can lead to a 'chicken-and-egg' problem - no one is willing to pay for the testing, unless there is some assurance that it will be properly done, but performing the testing is difficult unless funding is available.

Incompetency: A dedicated researcher is necessary - but alone, insufficient. He (or she) must also be competent. Unfortunately, the 'research' of several VTCs has in the past fallen short of this. The level of competence can generally be determined by viewing the virus collection quality - are there numerous non-viruses, droppers, and corrupted samples, or have all the files been 'weeded' and classified properly?

Obtaining a virus collection is easy - maintaining it is not. In some cases there has been an amazing neglect of fundamental issues. For example, when two samples contain the same virus, the fact that a particular scanner does not identify the samples as identical does not automatically mean that the test-sets contain different viruses: the scanner may be wrong. Conversely, even if a scanner claims two samples are identical, it does not mean that they contain the same virus. The only way to be certain about the contents of a test sample is to analyse it.

Another problem arises when the collection includes files which cause false alarms on a particular scanner. In this case, the scanner which is incorrect will look better (after all, it detects a virus!) than the other products which perform correctly and do not detect anything. A person who is not able to analyse virus samples in order to resolve problems like these has no business running a VTC.

Bias and Fraud: A VTC might rely on anti-virus producers for a steady virus supply. If this is the case, the collection will have a bias built into it which is difficult to remove. It is possible to allow anti-virus developers full access to the test-set, but that can also be abused - there are several cases of anti-virus products detecting only a few of the samples of a particular virus; just enough to detect all samples included in a specific test-set.

The other extreme would be to deny all access to the test-set, but then another form of abuse becomes possible: an unscrupulous anti-virus producer might construct a set of new viruses and submit them to a VTC, knowing full well that other anti-virus products will be unable to detect those viruses. The level of distrust which has existed in the anti-virus industry since the beginning, and the unwillingness of certain anti-virus producers to cooperate with some of their competitors, means that there can be no simple solution to this problem.

"no one is willing to pay for the testing ... but performing the testing is difficult unless funding is available"

Lack of Trust or Security: There have been cases where VTCs have allowed untrustworthy individuals access to a significant number of viruses. This may be a factor in the seepage of some viruses, supposedly available only to the research community, to the 'underground'. Often, this leakage was caused by lack of experience on the part of those people handling the viruses: some VTCs exchanged viruses freely with anyone willing to provide them with samples, when they discovered that many anti-virus developers are somewhat paranoid in this respect.

Unfortunately, some of the individuals who were freely willing to share their virus samples with the VTC turned out to be nothing but collectors who exchanged their samples with their favourite Virus Exchange BBSs.

This does not mean that it is impossible to get a VTC which can do a proper job. However, independent testing and anti-virus product comparisons are much more difficult than many who have attempted to enter that field seem to realise ■

The ABC of Virus Names

When naming a group of viruses which are closely related - i.e., they belong to the same family, have the same infective length and differ by only a few bytes - the usual practice is to give them names ending in A, B, C, etc. This works well for the first 26 variants, but what happens after Z? There are no more letters in the English alphabet, so a different approach is needed.

The problem was ignored until recently: unfortunately, the number of insignificant minor virus variants kept growing. For instance, at the moment there are 96 variants of Jerusalem.1808, 39 variants of Vienna.648 and 33 variants of Burger.560. Other viruses are now also approaching this limit - there are at present 22 1701 byte, and 20 1704 byte, variants of Cascade.

The solution was simple - after reaching Z, new variants will take a two-letter identifier, starting with AA and AB, and going up to ZZ. This gives 676 additional names, and, should that limit ever be reached, the obvious solution would be to use AAA-ZZZ.

Another new naming convention has been adopted this month in the IBM PC Viruses (Update) [*see p.4. Ed.*]. All virus names containing a space will now include an underscore in its place. This means that instead of the Dark Avenger virus, we will now have the Dark_Avenger virus. The benefits of this may not be obvious, but it resolves certain ambiguities. Also, writing a name like 'The_Rat' simply looks better than writing about 'the The Rat virus', at least from my personal point of view ▮

Identification versus Detection

The questions 'How many viruses does scanner X detect?' and 'How many viruses does scanner X identify?' seem very similar, but there is one major difference: it is possible to answer the second query but not the first.

Identification of a virus includes detection, of course, but also the ability to determine the identity of the virus. The number of viruses which can be identified can be determined (at least by the author) by counting the viruses to which they have access.

These are identified in different ways by the scanner, but it is impossible to determine exactly how many viruses the scanner will detect: an unknown variant could exist which the scanner might detect, and identify (incorrectly) as an older, already known virus. That unknown variant would then presumably not be identified correctly by the scanner.

Therefore, it is meaningless to claim, 'Scanner X detects 2345 viruses', but claims like 'Scanner X detects more than 2345 viruses' or 'Scanner X identifies 2345 viruses' are easy to defend. It is also possible to make more detailed claims; for example: 'This version of scanner X identifies 3249 different viruses.'

This still needs expansion: 'Of those, 1079 are identified exactly (i.e., even single-bit changes should be detected), but the other 2170 should be detected with sufficient accuracy to avoid corrupting them when disinfecting, because of misidentification. In addition, scanner X can detect viruses belonging to 193 other families, where no attempt is made to identify the viruses, giving a total of 3442.'

Unfortunately, detailed claims like this do not fit on a small sticker on the front of the package, but a phrase such as 'Detects 5000 viruses' or even 'Detects all known viruses' will - and looks impressive.

Speed and Accuracy

Four years ago Padgett Peterson said (and is still saying today), 'People need a fast *something* detector and a rigorous *what* detector.' (i.e. a very fast scanner, and an exact, not necessarily fast, identifier): I personally agree - however, there is no reason why those two detectors should be different products.

"independent testing and anti-virus product comparisons are more difficult than many ... seem to realise"

Most PCs are not usually infected with viruses. When they are being scanned for viruses, a fast scanner is obviously preferable - nobody wants to wait for an hour every morning while the hard disk is being checked. However, if the machine ever gets hit by a virus, scanning speed becomes a minor issue.

What really matters is whether the scanner can accurately determine the identity of the virus: this is an absolutely necessary condition for attempting disinfection. In other words, virus detection speed is important - virus identification speed is not. It therefore makes sense to try to optimise scanners to minimise the time spent on finding *if* there is a virus, even at the cost of a significant slowdown when a virus is found.

However, there is one small but important group of users who dislike this approach - reviewers who run scanners on their collection of samples, and complain about the speed. Unfortunately, what they do not seem to realize is that it does not make sense to check speed and accuracy at the same time. Instead of spending resources in trying to increase the identification speed, it would perhaps be better to try to educate the reviewers.

Virus identification has become as important as detection: as shown, they can be two completely different things. Paramount, even more so than speed in a scanner, must be identification - without this any anti-virus product must be considered useless ▮

TUTORIAL

Viruses on Unix Systems

James Beckett

To date, the world of the virus writer and anti-virus product author has revolved largely around the *IBM PC* running *MS-DOS*. Despite the inherent limitations of the operating system (single-user, single-tasking) and the grubby but necessarily backwards-compatible memory scheme of the CPU, the combination has become the *de facto* standard worldwide. *IBM's* release of PC specifications, and encouragement of third-party hardware, means that a program carefully written for the PC will run on any of millions of small, cheap machines.

This one fact makes the task of the virus writer possible - although many viruses are not written carefully, most will propagate unhindered through the vast user-base. Other business systems are, through their normal access control mechanisms, better protected against attack. However, such systems would be by no means immune if targeted.

PC viruses infect programs by adding code to a file in such a way that it will gain control when that program is executed. Under DOS, there is no way to prevent file modification - any running program has complete control of the system. All loaded programs can examine and modify any part of memory. Virus and anti-virus TSR play an escalating game of *Core Wars*, with total control of the PC as the goal.

Process Control

Multi-processing systems impose restrictions at a hardware level. Subject to appropriate system software, many CPUs (including the *Intel 386* and above) can run in User and Supervisor modes, where memory protection is available. Multiple applications run in User mode, and cannot access memory used by other programs or by the operating system, which runs in the privileged Supervisor mode and has exclusive access to the full resources of the system.

Multi-user systems require users to identify themselves by logging in. This identity is thereafter associated with all processes created on behalf of that user. File controls allow access based on file owner (the user who created it), the owner of the process requesting access, and permissions on the file. DOS has a minimal implementation of file permissions, designed to prevent accidental erasure rather than to implement security. The Read-Only flag is advisory: any DOS program, including a virus, can override this.

Unix Access Control

Under *Unix*, access control requires the user to supply an identifying name and a password before letting the user in. Then, file protections embody three kinds of controls for three sets of users. These are Read-permission, Write-permission, and Execute-permission. They apply respectively to the file owner (its creator), a designated group of users, and 'everyone else'. Typical file permissions might be:

- User may read or write, with no access to group or other;
- Read-write for user, read-only for group - e.g. sharing a report within a department but not allowing change;
- Read-write-execute for user, execute-only for group and other - a program which you wish to make available to all users, but of which you do not want to make multiple copies available.

All file permissions can be set or reset only by the file owner or the specially-privileged *root* account, used by the system supervisor. Disk access is arbitrated by Supervisor-mode operating system software and cannot be directly manipulated by applications.

Trust and Management

In order to infect a file, a DOS virus must find a writable executable and modify it. If all executables in a *Unix* system are not writable by the user, the virus cannot make the modifications required, because executed programs do not have complete control of the host machine. One fallback solution for the virus writer is to create new executables in writable directories in the hope that someone will run the program. The chance of this can be increased by naming the file after a common system utility like *ls* (list files) or *cp* (copy files), and by putting it in a directory known to be on users' search paths, such as /usr/games - sometimes left writable on relaxed systems.

"The ultimate coup would be to have the super-user root run an infected program, at which point the virus could do anything"

Any virus must either rely on such poor management and trust, or crack system security to get better privileges. Both approaches have no defence at all under DOS. No management can enforce write-protection if the system itself cannot, and 'DOS Security' is simply an oxymoron.

File infection is a transitive process, and under multi-user systems a new facet is introduced. When the virus has infected all it can under the permissions of one user, it can continue the process: another user may have access to files and directories the first did not. Thus, the virus gains privileges as it infects. The system permits users to trust each other, and is the basis for many system attacks.

How can one protect against this? Checks should be carried out to ensure that common directories are not world-writable, and checksums could be taken of standard utilities. Virus prevention is largely a question of good default options - the PATH should look in standard system directories first, and the current directory last, if at all.

Ultimate Power

A smart virus could check the level of privilege it has been accorded at each stage and use this as a trigger condition. The ultimate coup would be to have the super-user *root* run an infected program, at which point the virus could do anything, even compile a Trojan into the kernel or system utilities. Also, being accorded system privileges presents the opportunity to spread rapidly over network links, and to hide the virus' existence from other system programs.

In the news this month is just such a case, though by a user-controlled program rather than by a virus: having attained root privilege on a machine, the NIT (Network Interface Tap) facility was used to snoop on raw network data and intercept usernames and passwords as users logged in across a remote link. The information trapped can be used by a virus to log onto other machines itself, and propagate further.

Here, NIT is not really the security hole, as it is used by several system programs; accessibility of the network data is. A user with his own PC could snatch data in just such a way. Systems such as *Kerberos* are designed for system authentication and encryption of network traffic, and are available for several *Unix* varieties.

The superuser is all-powerful in a *Unix* system; a user (or virus) which gains this power has the run of the computer. Depending on local configuration, nearby systems may trust each other and so may also be compromised. However, the problems of trust are being addressed more rapidly, and many systems have a minimal dependency on each other without limiting legitimate use. Workstations are a weak point, so servers do not typically allow root on a workstation unlimited access to their own files.

Certain inbuilt system features can work for the virus rather than against it. *rdist* is a program designed for keeping a close group of similar machines up to date with system software, and certain directories on a designated server are checked for new copies of programs being added: these are automatically distributed to other machines in the group. If a virus were to get into such a directory on the server, the system would happily propagate it, with the other systems colluding and trusting the server. Conversely, if the virus found its way into the corresponding file on a client, it might be overwritten when the client was updated.

System Security

To make a real impact, a virus would have to embody knowledge of how to crack security on the targeted system. There are many bugs, or more usually configuration or management problems, which provide inroads to gain unauthorised power. Adept university students, and *Unix* professionals, have for years known of problems which, for one reason or another, the vendor has not fixed. Some of these people have a reputation for being able to crack the system at their leisure. The problems are being rectified, but if a virus could be embodied with knowledge of enough bugs, many systems would succumb to at least one.

That said, any such problem could be reactively fixed without disrupting system usability. Any worthwhile amendments to DOS to prevent viruses would either be circumventable or result in a system which simply was not DOS, and many legitimate working programs would break. Many *Unix* problems can be solved merely by changing access permissions or by simple kernel modifications.

File Infection

Infection of a *Unix* executable would proceed in much the same way as a DOS EXE file. The header information is available on most systems and is in some ways simpler than the EXE format:

```
/* format of the exec header
 * known by kernel and by user programs
 */
struct exec {
     unsigned long  a_magic;  /* magic number */
     unsigned long  a_text;   /* size of text seg */
     unsigned long  a_data;   /* size of initialized data */
     unsigned long  a_bss;    /* size of uninitialized data */
     unsigned long  a_syms;   /* size of symbol table */
     unsigned long  a_entry;  /* entry point */
     unsigned long  a_trsize; /* size of text relocation */
     unsigned long  a_drsize; /* size of data relocation */
};
#define OMAGIC  0407          /* old impure format */
#define NMAGIC  0410          /* read-only text */
#define ZMAGIC  0413          /* demand load */
```

A small virus could fit in the space between program segments, or, with more complex code to restore the state of the host's environment when done, add itself to the code segment. 'Magic numbers' at the start of the file indicate that the file is executable - *Unix* does not use a file extension to designate that a file is a program.

Depending on the processor and memory-management system used, it might not be possible for a program to read its own code segment or execute code from its data segment, and a virus might have to carry two copies of itself, or to resort to file accesses. Once again, this makes the job of the virus writer more difficult.

Architecture Limitation

Probably the biggest barrier to virus penetration of *Unix* is the diversity of system architecture. While the system call interface in a high-level language is clean, simple, and fairly constant between versions, compiled code may be very different, depending on the underlying hardware. A virus would have to target a particular hardware/system combination, or carry several versions of code. The same problem does not exist in DOS, as binary programs are designed to be run across different machines.

The *Unix* community distributes code in source form, which tends to help guard against infections by pre-compiled viruses. A virus could only be introduced by trojanised source code, which can be inspected before use.

The architecture problem can be largely sidestepped by using a shellscript program, which is an ASCII text file containing commands. Unlike the DOS batch system, there are powerful features for file manipulation, process control, expression calculation, etc. A simple virus can be written in shellscript alone, and experiments have shown such a virus to be reasonably effective. However, its code is immediately visible to anyone looking at the file. This is the only form of *Unix* virus yet seen, presented in a paper by Tom Duff of *AT&T* in *Computing Systems*, Spring 1989.

Boot Processes

To cover fully the possibility of virus infection, the *Unix* boot process should be considered in the same way as DOS. It is worth noting the rise of PC-based *Unix* platforms, and drawing a distinction between DOS viruses and BIOS boot viruses. A DOS virus relies on the file services of DOS to propagate, whereas Master Boot Sector viruses only rely on the PC's system-independent boot process to gain control, and the BIOS to propagate.

"having attained root privilege on a machine, the NIT facility was used to ... intercept usernames and passwords"

Once its boot process is complete, DOS uses the BIOS for its disk accesses: thus a boot virus spreads to other floppies. If a *Unix*-based PC is booted from an infected disk, the virus has immediate control, and could trigger - Michelangelo, for example, would happily destroy a *Unix* hard disk.

From then on, a virus designed for BIOS/DOS is in jeopardy: as *Unix* provides its own disk routines and ignores the unsuitable BIOS ones, the virus will not spread. When *Unix* starts up, the virus will become a fish out of water.

In a conventional *Unix* machine, the boot process starts in the EPROM program provided by the manufacturer. Any device can be specified for booting: open tape, cartridge, floppy disk, hard disk, or network are typical options. Usually this is set in hardware to point to the hard disk, although this can often be overridden. A bootable floppy disk would be an easy way for a virus to gain entry on a workstation, though on a non-PC-based *Unix* it would usually require collusion on the part of the user.

When booting from the hard disk, the disk boot program loads in the first sector of the disk to a fixed point in memory and hands control to it. This then loads several more sectors of boot program from the root partition; enough to understand the basic structure of the boot disk. The kernel file '\Unix' can then be loaded and run. This is the heart of the system, implementing all the process control and file access mechanisms. Modifying the kernel would be a potential infection technique, but would require much machine-specific knowledge.

Many system programs are then started and left running in the background to control and monitor the system, and these usually have root privileges. Any of these are potential targets for a virus.

All these areas should be protected against modification. The system files should only be modifiable by root, and any way of gaining unauthorised root privileges should be prevented.

The end of the chain is the user command processor, or shell. This accepts input from the user and initiates the running of requested programs. There are many situations where a program other than the one the user intends can be run, e.g. the PATH problem of writable directories discussed above. Privileged programs can also be tricked into doing things they should not, or giving the user higher privileges.

Prevention and Detection

Tom Duff proposed a countermeasure involving changing the meaning of the 'execute' flag on files. This historically means simply that a program has been fully compiled and linked: 'permission rather than certification'. Duff suggested that an authority should examine all files to check that they should be accorded certification, and that any modification of a certified file would lose it its certification. This process seems not yet to have been implemented.

Other kernel changes could implement a checksumming mechanism to ensure that files are unchanged from a certified 'clean' state. Both would impede someone developing new software unless they are given a special 'certify' program, the use of which could be logged.

Conclusion

In summary, it can be seen that there are a number of factors which limit the spread of computer viruses on the *Unix* platform. The principal hurdle a potential virus writer must overcome is that posed by the security built in to *Unix*: a process runs with the same privileges as the user who created it. Under DOS, any program is effectively root.

Secondly, the vast array of different hardware configurations on which *Unix* runs forces any successful virus either to limit the machines upon which it will function, or spread in a high level language form, be that shellscript or C. Once again, this makes the process of writing a *Unix* virus vastly more involved than the 100-byte programs found in DOS.

Although *Unix* has, by and large, escaped virus problems to date, that situation may change in the future. Even with the difficulties outlined above, it is possible to write a *Unix* virus; it seems likely that such an undertaking will be highly attractive to a largely *Unix*-based computer underground.

PRODUCT REVIEW 1

CPAV for NetWare

Jonathan Burchill

Central Point's Anti-Virus for NetWare (*CPAVNet*) claims to provide a complete operating environment for virus detection and protection. The package is designed to protect both the file server (or servers) and DOS workstations, and provides a message management and alert system. File server-specific protection is provided by NLMs (*NetWare* Loadable Modules), whilst DOS, *Windows* and *Macintosh* workstation protection is provided by a mixture of workstation-based scanners and anti-virus TSRs.

Central Point claims to have increased the functionality of *CPAVNet 2.0*, and the product now contains a number of new features, including *Central Alert*, a server-based package which generates alert signals whenever the server or a workstation detects a problem, and *Central Setup*, which can force users to run *Central Point's* workstation packages. *CPAVNet* sounds like an MIS Manager's dream. Does it live up to this in the cold light of day?

Installation

The package contains versions of the software for different platforms (see below), a well-indexed manual, and assorted bumph. Overall, the quality of the documentation is excellent in terms of installing, configuring and using the software. It does, however, lack any real detail on the processes involved on detection and protection, making it difficult to assess its true strengths and capabilities.

The package includes *CPAV for DOS*, *NetWare*, *Macintosh* and *Central Alert*. The DOS and *NetWare* software is supplied on both 3.5-inch and 5.25-inch media. I was not able to try the *Macintosh* software, so the remainder of this review concerns only the DOS and *NetWare* components.

Most of the programs are supplied in both DOS and *Windows* versions. The functionality of the two is identical, and it is really a matter of preference which client is used. All installation and configuration of the NLM is carried out from a DOS workstation. Thus, before installing the NLM, it is necessary to install the DOS software and check that the workstation is clear of viruses.

The NLMs require that the server has at least 8MB of RAM and is running *NetWare 3.11* or later (including *NetWare 4.0x)*. No support is offered for older 286-based file servers. *Central Point* allows several servers to be protected at once. To do this, the protection NLM is loaded on every server, whilst a master NLM and the optional *Central Alert* NLM is loaded onto the 'Master server'. This ability to link several servers into one or more groups is extremely useful, allowing centralised control of multiple servers.

When installing the server software, the user must be logged in as Supervisor. This condition satisfied, the install program copes with all the configuration details, and optionally modifies the server configuration to load the protection NLMs automatically on powerup.

Once installed, the product is very modular. I shall therefore consider each part of the product in turn, before going on to discuss the virus detection capabilities of the system.

CPAVNET: Operation and Options

Although each server displays a control screen on the system console, the package is designed to be controlled and configured from a workstation, using the CPAVNET program. As stated earlier, both the DOS and *Windows* versions of CPAVNET offer identical facilities, with almost identical interfaces. The DOS version offers the now standard *Central Point* DOS text GUI (if such a concept exists), allowing both keyboard and mouse control.

When using CPAVNET, the user is in effect logged into a domain (defined at installtime) of servers, within which it is possible to select and configure individual servers. This centralised control facility is the concept which will allow virus prevention for an entire company to be carried out from one terminal.

Two types of virus-specific detection are offered, together with two methods of checking. File detection is based on both looking for known virus signatures in files and optionally using the 'virus analyser' software, which looks for suspicious instruction sequences. File checking may be carried out as either real-time, scheduled or both. Real-time checking can be set to monitor incoming and/or outgoing files for any combination of DOS and *Macintosh* viruses.

The list of files scanned can be configured by extension, by exception list, and by inclusion list. *Central Point* recommends that the virus analyser overhead is not incurred in real-time scanning. However, on my trial server, it added such little overhead that I would seriously consider changing this option from the default setting.

Scheduled scanning can be either periodic (e.g. Monday and Friday at midnight), or interval based, such as every 2 hours. On completion of a scheduled scan another NLM (for instance a backup NLM) may be loaded and executed.

CPAVNET also allows several actions to be taken on discovering a virus. These include specifying to whom an alert message should be sent (the file owner, the last updater of the file etc.) and what to do with the infected file. Options for infected files include doing nothing, deleting the file or moving the file to a pre-designated directory. Moving the file is an excellent option (especially for product reviewers!) as

the directory to which it is moved can be set to have no public access rights, thus removing the file from further access whilst retaining it for later analysis.

The results of NLM activity and reports from the workstation software (see below) are sent to a server-based log file, which grows at an alarming rate, and can quickly become enormous. Therefore, *Central Point* has included various options for purging of old records and filtering the contents to show specific information.

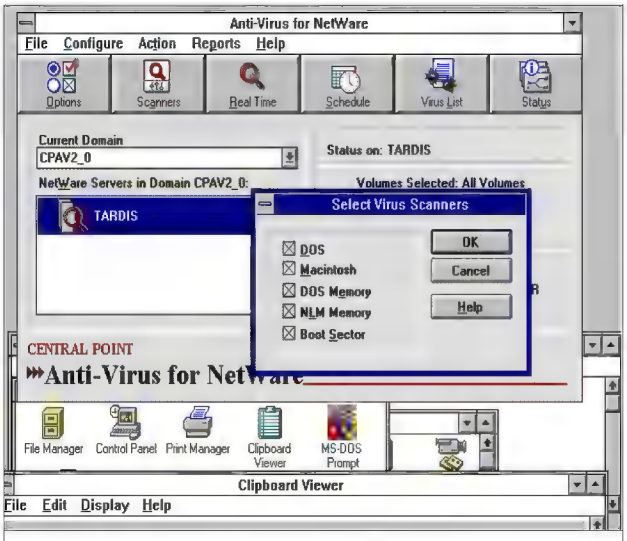
One major gripe is that no information is provided on the structure and type of information in the log file, thus preventing third party report generators from being used. This is inexcusable and should be remedied as soon as possible.

Central Setup

Central Setup automates the installation and configuration of the DOS workstation software, and adds a CINSTALL line to the server SYSTEM login script. As a user logs in, *Central Setup* is able to verify that the user is running the specified protection TSRs, copy new ones to the workstation if they are either missing or out of date, check the WIN.INI and AUTOEXEC.BAT configuration files and reboot the user's workstation if required, so that the changes are forced to happen.

It is possible to exclude logins until workstation software is correctly installed and active. *Central Setup* also offers configuration of the time period between workstation scans.

These options apply to groups of users. This grouping is the same as the standard *NetWare* groups as configured on the file server. I am not convinced this is the best way to classify users: it is possible to except an individual from the group requirements but not possible to set specific requirements for an individual (only for the group to which he/she belongs). I would have thought it more likely that protection requirements should be set according to the hardware of the specific workstation (unlikely to be reflected in the *NetWare* groups), and standard *Novell* security used to control who logs in from where.



The principal strength of *CPAVNet* is its excellent control facilities, allowing the entire network to be configured from one workstation.

Central Alert

Central Alert is an NLM which can receive messages from *Central Point* packages and then carry out a predefined alert action. The packages which send messages include *CPAV for NetWare*, *DOS* and *Macintosh*, *DiskFix*, *Build Emergency Disk*, *Disk Optimizer*, *VSafe* and *Vwatch*. Alerts are on one of five levels of severity, and at each level may be sent to any combination of Electronic pager (both alphanumeric and numeric), *NetWare* broadcasts to specific workstations, a log file, MHS Email, or SNMP messages.

Like the scanner log file, *Central Alert* allows the log file to be filtered and displayed in a number of ways. Once again, no details of the log file structure is given.

CPAV for DOS

Workstation protection is provided by *CPAV Version 2.0 for DOS*. As this has been reviewed in *Virus Bulletin* in full (*VB*, August 93, pp.16-19), I will limit myself to a brief overview of the facilities, noting the changes which occur when this package is run on a protected network. Protection consists of a number of components. Firstly, there is *CPAV*, the *Central Point* file scanner and immunizer, then the TSR utilities *Vsafe* and *Vwatch*, which look for virus activity, *Bootsafe*, a utility which helps guard against boot sector infection, and a scheduler, to configure unattended scans.

The scanning utilities, like the *NetWare* components, provide the same detection options as the NLM. In addition, the scanner is able to look into files compressed using *PKLite*, *PKZIP*, *LZEXE* and *ARJ*. The workstation software is also able to compare the file against an integrity database built from the original file's date, time, size, attributes, and checksum. No information is given about how the checksum is calculated, or how reliable it is.

The scanner will attempt to clean an infected file, and also allows files to be 'immunised' against infection - this consists of adding circa 1K of extra code to the executable, supposedly to help the executable discover it has been infected and heal itself. I find this procedure somewhat dangerous, and am rather sceptical of it.

The effect of running the software within a protected file server domain is twofold. Firstly, if *Central Setup* has been used, the configuration and operation of the software on the workstation is checked at every login. Secondly, various components of the workstation software can send alerts and status reports to the file server.

As several reviewers have already remarked, *CPAV* hangs the machine if it discovers a lot of infected files. This was either a complete crash, or various error messages such as

'fatal error writing drive c !'. Playing with configuration options altered the point and nature of the crash, but I was never able to run the software without it hanging. In the end, I did get some results by presenting the Standard test-set to the scanner in small sections. This was a complete pain, and is unacceptable in the real world. Come on *Central Point* - fix the problem!

Detection Results

As the test results show, the scanner is passable on both the Standard and the In The Wild test-sets. The NLM missed 19 samples from the In The Wild test-set, including Satanbug and Todor, and all five samples of Tremor - not very inspiring results, but not disastrous either. When run against the Standard test-set, the results were more encouraging, with a total of only 15 missed files.

The results from the newly expanded Polymorphic test-set were also not as good as they could be: *CPAV* missed 87 samples from both the Mutation Engine and Uruguay.4 sections of the test-set.

The workstation software similarly in the detection results, finding 213 infected samples in the Standard test-set. This rose to 219 with the virus analyser turned on. Comparing the results for the signature detector and the virus analyser showed that the analyser found about 70% of the viruses which the signature detector found, plus a handful which it did not. Thus the virus analyser provides some level of additional protection against unknown viruses.

As a final test, I took three viruses the scanner had missed - Todor, Pitch and Power - and executed them. Only Pitch produced complaints from *Vsafe* as it attempted to modify the executable, memory and COMMAND.COM. Todor and Power were allowed to execute without complaint, though subsequent scanning with *CPAV* showed several files as infected with an 'unknown virus' (Oh well, off to find the clean boot disk). Given that the *NetWare* scanner did not score 100%, it would be possible to spread an infection fairly widely before it was detected. Certainly, this shows the importance of regular use of a scanner and a checksummer.

The question of overheads when scanning under *NetWare* is a tricky one, which cannot be fully addressed without some comparative results. I will therefore hold fire on this aspect of the scanner - a full NLM comparative review will be compiled in the summer. Suffice to say that I found the overheads noticeable but acceptable.

Conclusion

Overall, the integrated environment of server *Central Talk* and *CPAVNet* works well. The virus management features provided are excellent, but the virus detection results are disappointing - as such, it will appeal to those sites where the real problem is security management. *Central Point* has many of the features of a top-class utility, but in order to beat its competitors, detection rates must improve.

CPAV for NetWare

NLM Detection Results:

Standard Test-set[1]	212/227	93.4%
In the Wild Test-set[2]	90/109	82.6%
Polymorphic Test-set[3]	338/425	79.5%

Scanning Speed:

Speed results for an NLM product are inappropriate here, due to the multi-tasking nature of the operating system. Full comparative speed results and overheads for all current NLMs will be printed in a forthcoming VB review.

For full details of the DOS workstation product see Virus Bulletin, August 1993, pp.16-19.

Technical Details

Product: *Central Point Anti-Virus for NetWare*

Version: 2.1

Vendor (UK): *Central Point Software*, 3 Furzeground Way, Stockley Park, Uxbridge, Middlesex UB1 1DA.
Tel. +44 (0)81 848 1414, Fax +44 (0)81 569 1017

Manufacturer: *Central Point Software*, 15220 NW Greenbrier Parkway, Suite 200, Beaverton, OR 97006-5764 USA.
Tel. +1 503 690 8088, Fax +1 503 690 2688

Price: £699 + VAT (RRP for server and unlimited workstations)

Hardware Used: Server - 50Mhz 486DX, 16 MB ram, 300Mb disk space, Client - 33Mhx 486DX, 8Mb ram, 180 Mb disk space.

The viruses used for testing purposes have been updated this month. Each test-set contains genuine infections (in both COM and EXE format where appropriate) of the following viruses:

[1]**Standard Test-Set:** As printed in *VB*, February 1994, p.23 (file infectors only).

[2]**In the Wild Test-Set:** 4K (Frodo.Frodo.A), Barrotes.1310.A, BFD-451, Butterfly, CAPTTRIP, Cascade.1701, Cascade.1704, CMOS1-T1, CMOS1-T1, Coffeeshop, Dark_Avenger.1800.A, Dark_Avenger.2100.DI.A, Dark_Avenger.Father, Datalock.920.A, Dir-II.A, DOShunter, Eddie-2.A, Fax_Free.Topo, Fichv.2.1, Flip.2153.E, Green_Caterpillar.1575.A, Halloechen.A, Helloween.1376, Hidenowt, HLLC.Even_Beeper.A, Jerusalem.1808.Standard, Jerusalem.Anticad, Jerusalem.PcVrsDs, Jerusalem.Zerotime.Australian.A, Keypress.1232.A, Liberty.2857.D, Maltese_Amoeba, Necros, No_Frills.843, No_Frills.Dudley, Nomenklatura, Nothing, Nov_17th.855.A, Npox.963.A, Old_Yankee.1, Old_Yankee.2, Pitch, Piter.A, Power_Pump.1, Revenge, Screaming_Fist.II.696, Satanbug, SBC, Sibel_Sheep, Spanish_Telecom, Spanz, Starship, SVC.3103.A, Syslock.Macho, Tequila, Todor, Tremor (5), Vacsina.Penza.700, Vacsina.TP.5.A, Vienna.627.A, Vienna.648.A, Vienna.W-13.534.A, Vienna.W-13.507.B, Virdem.1336.English, Warrior, Whale, XPEH.4928

[3]**Polymorphic Test-Set:** The test set consists of 425 genuine samples of: Coffeeshop (375), Uruguay.4 (50).

PRODUCT REVIEW 2

Virus Check & Cures

Dr Keith Jackson

I thought I kept up to date with various PC anti-virus products, but I must admit I had never heard of *Virus Check & Cures* until I received this review copy. The product is supplied as a single 1.44 Mbyte floppy disk (3.5-inch, non-write-protected), a small booklet, and a few pieces of bumph. Packaging is excessive - at least 30 copies of *Virus Check & Cures* could fit in the product's box. However, one cannot tell a book by its cover; does *Virus Check & Cures* live up to the maxim 'small is beautiful'?

Documentation

The documentation comprises a small booklet (8 pages of A6), not unlike those which accompany various non-prescription medicines, and is about as much use. This tiny pamphlet (use of the word 'manual' would imply something far more grandiose) explains component parts of *Virus Check & Cures*, and how to install them. The booklet states that '*Virus Check & Cures* calls for minimal use of this users manual' - a good job that it does! Reference is made to available on-line help files, but these are virtually useless at providing technical detail. One contains this memorable phrase: 'Refer to user manual for more information on features not listed above and other *Virus Check & Cures* options.' Don't bother.

Fifteen text files provided on disk explain how the various 'cure' (their word) programs included with *Virus Check & Cures* work. These are all similar, but vary in detail on each virus. Most warn that when a certain virus is 'cured', there is a 'small risk of loosing host'. Even allowing for the nonsensical English, I am not sure what this means.

One clause in the software licence really grabbed my attention: it explains clearly that the developer of *Virus Check & Cures* is responsible for nothing whatever, even if he 'has been advised of the possibility of such damages'. Only a lawyer could concoct such a phrase. In these litigation-hungry times, software vendors commonly disclaim all responsibility for their wares, but given some of the instructions included with *Virus Checks & Cures*, this additional disclaimer is probably well-advised.

Installation

Two installation methods are provided, one each for DOS and *Windows*. They do basically much the same thing, initialising options and copying files to the hard disk: both versions of the main executable file are in fact one and the same. Although it takes an age to copy across all files required by *Virus Check & Cures*, the DOS installation program works well. Amongst many available options is the choice of subdirectory into which the files will be placed, and the ability to make alterations to AUTOEXEC.BAT.

The *Windows* installation program is poorer than its DOS cousin. When the *Virus Check & Cures* files are copied to hard disk, they are tagged with the date and time of installation, rather than retaining the original values. The file DERBCURE.DOC is installed with size 0 bytes (and no warning message is produced).

Virus Check & Cures cannot make its icon appear until *Windows* has been rebooted - something which other packages are capable of doing. The *Windows* installation program also insists on using a fixed subdirectory location for all its files (C:\VP) - the C:\VP prefix is hard coded many times within the program. I place files on my hard disk where I want them, and I refuse to use software which insists on a fixed subdirectory location.

"The documentation claims that Virus Check & Cures detects '99.9% of disk errors': it seems to detect over 100%!"

Features

Virus Check & Cures has five components: an investigation program (actually a checksummer), a virus scanner, an audit program, a memory scanner, and several 'cure' programs. If alterations to AUTOEXEC.BAT have been activated during installation, a memory-resident program occupying 16 Kbytes of RAM is added when the PC is next booted: this provides an audit of computer activity.

The checksum program is also executed from within AUTOEXEC.BAT to ensure that files on the hard disk have not been altered. When I tested this, *Virus Check & Cures* produced an error message saying, 'An error has occurred trying to access -> C:\vpvpar <- Disk File'. Neither this, nor any other error message, is explained (or even mentioned) in the documentation. I eventually deduced that the checksum program was complaining that it had no checksum information to verify. The cure for this proved to be to take a 'snap' (their word) of the hard disk, and reboot. Why did the installation program not do this itself, and why does nothing either onscreen or in the documentation apprise the user of this fact?

The main features of *Virus Check & Cures* are accessed from an executable program offering drop-down menu access to the features, including viewing audit trail files created by the memory-resident monitor.

Memory Scanner

One component of *Virus Check & Cures* provides the facility to scan memory for viruses. This program maintains an onscreen count of the total amount of memory scanned while it is executing. This counter stopped at 1080 Kbytes, having taken 3 minutes 19 seconds to scan this amount of RAM. The memory counter went on to 1087 Kbytes when executed under *Windows*, and memory scan time rose to 3 minutes 45 seconds. Either way, this corresponds to just over 5 Kbytes per second, a figure which would win no prizes for speed. Apart from rebooting, execution cannot be terminated once a memory scan has commenced. Scanning of memory stopped at 1.08 Mbytes when five Mbytes are actually installed - no explanation as to why is provided.

Disk Scanning

When scanning the hard disk of my test computer, *Virus Check & Cures* took 6 minutes 55 seconds to search through a hard disk containing 15 Mbytes of files. When the same disk was scanned under *Windows*, the scan time rose to 10 minutes 17 seconds. By default, *Virus Check & Cures* scans for EXE, COM and SYS files; it inspected 331 files out of 978 present on the disk. This is slow - in comparison, *Dr. Solomon's Anti-Virus Toolkit* did the same in 1 minute 24 seconds, and *Sophos' Sweep* carried out a 'quick' scan in 2 minutes 15 seconds. Both programs searched over 400 files.

One confusing feature of *Virus Check & Cures* is that, although a 'Fast Scan' option can be activated, my timing measurements showed that this had no effect on scan time. Eventually, I realised that despite use of the word 'scan', the option referred to is the checksumming component of *Virus Check & Cures*. The documentation contains no help or explanation of this point.

The scanner always found disk errors when executed. When I tested *Virus Check & Cures* against the virus test-set described in the *Technical Details* section, it reported a disk error on 51 different occasions - but there is nothing wrong with my hard disk. These errors were reported while carrying out scanning tests against 238 different virus samples, corresponding to a false reported error rate of 25%. This is completely unacceptable. The most ludicrous error reported was 'Disk sector not found, this may not be critical'. I view such a problem as very critical indeed.

For the record, the normal *Virus Check & Cures* termination message is always 'Disk Error Investigation Completed'. Does this tell the user that testing has been completed successfully, or has an error been found? I haven't a clue. The documentation claims that *Virus Check & Cures* detects '99.9% of disk errors': it seems to detect well over 100%!

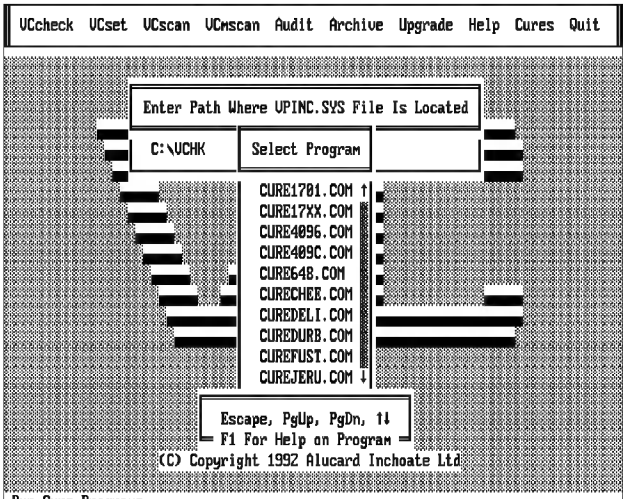
Virus Check & Cures refused to access the virus test-set when it was stored on a magneto-optical drive, continually asking for the write protection tag to be removed. The chances of that happening with my standard test-set are zero. To test the product's detection capabilities, I had to resort to copying files across to the hard disk, where the message 'Infected File Archived and Locked' was displayed whenever a virus-infected file was detected. The explanation of this message is startling, and deserves discussion.

Words fail me here, so I shall let the documentation speak for itself: 'Floppy disks being scanned must NOT be write protected, as VCSCAN requires to read hidden, system and read only files. To accomplish this, the file attributes are switched off and on while being read.' This is a red herring, as all scanners I have ever met scan hidden files without needing to do this. However, *Virus Check & Cures* really does refuse to scan a disk unless write-protection has been disabled. Imagine the consequences: a virus infection is known to exist at a particular company, and all floppy disks must be scanned to find out which ones are infected. *Virus Check & Cures* insists on removal of write-protection, thus running the risk of every floppy disk becoming infected.

Virus Check & Cures freely uses the term 'inoculation', but does not explain its meaning anywhere. After digging around, and finding a hidden subdirectory in the root directory of my hard disk, I realised that when a file was thought to be infected, it was moved to the hidden subdirectory, and replaced by a small executable file, renamed with the same name as the original. When executed, this file produces a warning saying it has been 'disabled by *Virus Check & Cures*'. The error message which prompted the discussion above now begins to make sense: an infected file is 'archived' by moving it to another subdirectory, and 'locked' by being replaced by another program.

Virus Detection Capability

Virus Check & Cures failed to detect 42 of the virus test samples listed in the *Technical Details* section, corresponding to an 83% detection rate: very poor. It performed noticeably better against older files in the test-set, failing to detect 9 of the 179 files in the original *VB* test-set (a 95%



Virus Check & Cures provides a selection of 'Cure' programs for optional removal of viruses from files and boot sectors.

detection rate). Test samples added in December 1993 went completely undetected, and of the files added in the previous test-set update (about a year ago), only 50% were detected. No Mutation Engine samples were detected. No boot sector viruses could be used as test samples, as I refused to remove write-protect tags and run the risk of *Virus Check & Cures* fouling up future testing. By then, my patience had gone beyond copying each boot sector virus to another floppy disk, merely to satisfy the stupid write-protect tag constraint.

When a file is found to be infected, one of the 'cure' programs provided with *Virus Check & Cures* can be used. These search the entire hard disk for a specific virus, and then attempt to remove it from any executable file found to be infected. I am not in favour of such tactics; however, *Virus Check & Cures* does state that replacement of an infected file with an original uninfected copy is the best option. The user must choose the correct cure program; *Virus Check & Cures* does not do this automatically. Given detection of a particular virus, why not?

The following quote from its documentation provides some insight into *Virus Check & Cures*: 'You will notice that when VCSCAN is run it takes longer than other common virus scanners. This is because we feel that the theory used by most virus scanners may be incorrect and that a more thorough check is required to pick up new strains of existing viruses.' *Virus Check & Cures* is slower than other scanners; it is also out of date, and extremely poor at detecting viruses. As for their theory that other scanners are not following the one true path which leads to virus detection, this is not borne out by the facts. *VB* prints comparative reviews showing which scanners are best at virus detection - and *Virus Check & Cures* is one of the worst.

Checksumming

I have already pointed out that *Virus Check & Cures* installs a checksumming program into AUTOEXEC.BAT, executed every time the PC is rebooted. It has two modes of operation: Slow and Fast. The former seemed to detect every single bit change made to any executable file. Fast seemed only to detect changes made near the start of a file. The checksum information was generated by taking a 'snap' of the hard disk, a process which took 1 minute 3 seconds on my test computer. *Virus Check & Cures* took 2 minutes 5 seconds to verify these checksums in Fast mode, and a whopping 14 minutes 54 seconds in Slow mode. Addition of either of these times to normal bootup time is, frankly, unworkable.

False Claims and Strange Messages

After all this, I was astounded to see that the developers had the gall to claim that: 'There will always be new viruses appearing, so *Wizardworks* will endeavour to update VCSCAN on a regular basis'. The latest files on the review copy provided are dated 15 July 1992: perhaps regular, but certainly not frequent. An interval of over eighteen months between updates is completely unacceptable, and I fail to see how their claim stands up to scrutiny.

Virus Check & Cures claims that it requires DOS v3.30 or above because 'DOS 3.2 had several unstable problems'. *Microsoft's* lawyers would surely be pleased to hear this. The text file JERUCURE.DOC, included with *Virus Check & Cures*, states that the original Jerusalem dates from 1984: in fact, it is later, circa 1987/88. The program CUREDELI.COM (which removes Joshi from an infected disk) contains the text string 'Remove any write protect tab from your Backup'. This is daft. Take a copy, experiment by all means, but removing a write-protect tab from a backup and letting a program loose on it is asking for trouble.

A final piece of nonsense: 'no virus detection systems tested up to present can identify an infection of Deli!' (i.e. Joshi). This is not, and cannot be, true - if the precise details of a virus are known, it can always be detected.

Conclusions

This program should be put out of its misery and have its floppy disk snapped in half. It has the feel of an unpolished product, yet the files are old. Did the developers give up? It is slow at scanning, so infrequently updated that the scanner is very out of date, so slow at checksumming that it is, in practice, unusable. It installs files in a fixed location when *Windows* is used, produces spurious disk errors, and has no meaningful documentation. This could be the first product I have reviewed where users might be better off doing nothing than following the stated (and insurmountable) instructions to remove write protection from floppies before scanning.

I would not use *Virus Check & Cures* under any circumstances, and would not recommend it to my worst enemy. Don't buy it.

Technical Details

Product: *Virus Check & Cures*

Developer: *Wizardworks*, 5354 Parkdale Drive #104, St. Louis Park, MN 55416, USA. Tel. +1 (612) 544 8581. Fax number not supplied.

Vendor: *Software Partners Publishing and Distribution Ltd*, Unit A, Meadow Lane, St Ives, Cambs PE17 4LG.
Tel. +44 (0)480 497622, Fax +44 (0)480 493614.

Availability: *IBM* or compatible with DOS 3.3 or above and (optionally) *Windows 3.0* or higher. 512 Kbytes of RAM required. *Novell, 3-COM, Artisoft* and *Token Ring* networks all supported.

Version evaluated: 2.00, 2.00.386, 2.198, 3.00 or 3.00.86 (depending on where you look)

Serial number: None visible

Price: £19.99 (excluding VAT)

Hardware used: A *Toshiba 3100SX* laptop, incorporating a 16 MHz 386 processor, 5 Mbytes of RAM, one 3.5-inch (1.4 Mbyte) floppy disk drive, and a 120 Mbyte hard disk.

Viruses used for testing purposes: A suite of 158 unique viruses (according to the virus naming convention employed by *VB*), spread across 247 individual virus samples, is the current standard test-set. A specific test is also made against 1024 viruses generated by the Mutation Engine (which are particularly difficult to detect with certainty).

For a complete listing of the viruses in the test-sets used, see *Virus Bulletin*, February 1994, p.23.

SUBSCRIPTION RATES

Subscription price for 1 year (12 issues) including first-class/airmail delivery:

UK £195, Europe £225, International £245 (US$395)

Editorial enquiries, subscription enquiries, orders and payments:

Virus Bulletin Ltd, 21 The Quadrant, Abingdon, Oxfordshire, OX14 3YS, England

Tel. 0235 555139, International Tel +44 235 555139
Fax 0235 559935, International Fax +44 235 559935

US subscriptions only:

June Jordan, *Virus Bulletin*, 590 Danbury Road, Ridgefield, CT 06877, USA

Tel +1 203 431 8720, Fax +1 203 431 8165

END NOTES AND NEWS

The winds of changes are blowing at *McAfee Associates*. Company founder (and ex-CEO) John McAfee has recently resigned from the post of Chief Technical Officer in the company after only four months. He remains Chairman of its Board of Directors. Also, 2,000,000 shares of the company's common stock, were snapped up at a 'significant discount' by *Summit Partners* and *TA Associates*. This was previously held by the J&J Trust, of which John McAfee is a beneficiary. In a separate move, the company has signed a 'non-binding letter of intent' to buy the assets of *Brightwork Development*, a network management software supplier.

Intel and *Novell* have decided to make their respective network management tools work together, and are 'tightly integrating' *Intel LanDesk Virus Protect* into *NetWare's* new *NetWare Distributed Management System*. What this actually means remains to be seen.

A one-day seminar on the forensic examination of personal computers, aimed at internal auditors and security managers, will be held at *Hambro's Conference Centre*, London, on Friday 8 April, 1994. For further information contact Rachel Forrest. Tel. +44 (0)71 344 8100.

The *VB 94* Conference will be held on 8-9 September 1994, at the Hôtel de France, Jersey. Tel. +44 (0)235 531889.

The US *Department of Defence* has announced plans to limit military links to the Internet. *Network World*, a computer magazine, reports that defence officials intend to add a protective relay to the worldwide Defence Data Network (also known as *Milnet*). Fears that hackers could invade highly sensitive data networks have apparently sparked the proposals, which have as yet to be implemented.

Central Research Laboratories have announced the launch of a fingerprint identity verification system, known as Verid. It works by digitally scanning fingerprints, which are then stored. When the fingerprint is re-presented to the scanner, the stored data is analysed and cross-referenced to verify identification. Tel. +44 (0)71 388 9988.

Central Point has released a new version of its *PC-Tools for Windows* product, which includes *CPAV for Windows*. The product is available at a special introductory price of just £69 + VAT. Tel. +44 (0)81 848 1414.

RG Software has announced a new UK distributor, *Hi-Tech Marketing Services Ltd*, for its anti-virus product *Vi-Spy*. For further information, contact Ken Highland. Tel. +44 (0)61 941 5073.

The 5th European Forum on IT Quality and Security, organised by *XP Conseil*, will be held on 16/17 March 1994 at the Hôtel Concorde St Lazare in Paris. *EUROSEC '94* will address, amongst other topics, the future developments and needs of IT security. Further information from Mme Hachin on +33 1 42 68 17 16, fax +33 1 42 66 22 56.

According to a report in *International Banking Regulator*, **US spy agencies may be tapping foreign banks' computers**. The report claimed that the US *National Security Agency* (*NSA*) had secretly implanted a 'trap door' into a Trojan Horse version of a VAX program named PROMIS, which gave the *NSA* the ability 'to directly access computers running the program'.

Hackers Beware. Malcolm Farquharson was sentenced to six months imprisonment for UK *Computer Misuse Act* offences even though he had never touched a computer! Farquharson made telephone calls to Emma Pearce, an employee of *HL Communications*, and obtained confidential information from the company's computer.

Patricia Hoffman's *VSUM* ratings for January: 1. *Command Software's F-Prot Professional 2.10g* - 97.0%, 2. *Safetynet's VirusNet Pro 2.10* - 95.2%, 3. *McAfee Associates ViruScan v111* - 94.6%, 4. *Dr. Solomon's AVTK 6.55* - 88.4%, 5. *IBM Anti-Virus/DOS 1.04* - 87.0%. **NLMs:** 1. *McAfee NetShield (3.11) 1.56v111* - 95.5%, 2. *Dr. Solomon's AVTK NLM 6.54* - 84.2%, 3. *Command Software Net-Prot 1.22* - 79.0%, 4. *Cheyenne's InocuLAN 2.0/2.18g* - 62.5%, 5. *Intel LanProtect 1.53+8/93S* - 59.1%, *Central Point AV/NLM 1.0* - 47.3%.